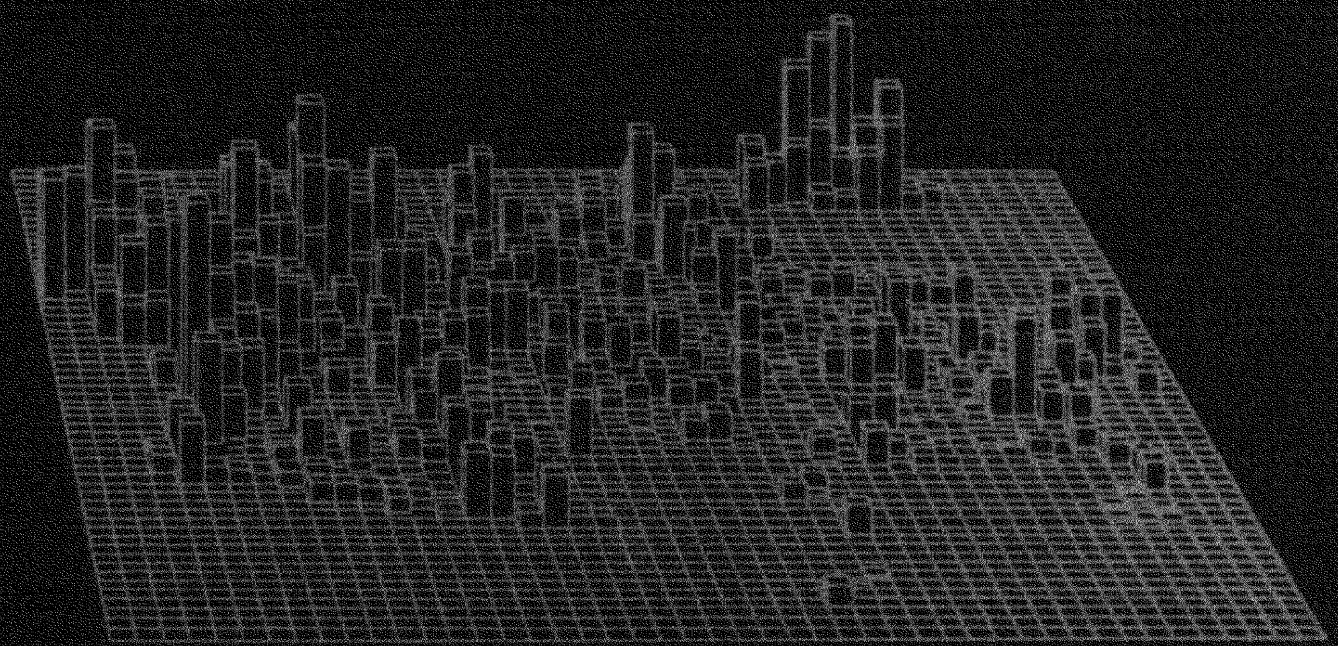
CRU

Storage of the 1971 U.K.Census data; some technical considerations

by M.Visvalingam



Census Research Unit
Department of Geography
University of Durham

Working paper 4

The Census Research Unit, Department of Geography, University of Durham, is a small group of research workers investigating aspects of the theory and use of census data. It is currently funded as a research project by the Social Science Research Council.

The diagram on the cover represents total population per 1 km grid square in the northern part of County Durham: the height of each column is proportional to the population in that square. The county is viewed from the west, Gateshead being at the extreme left margin, West Hartlepool at the far right and Bishop Auckland at the centre-right. The original surface was calculated and drawn by computer.

UNIVERSITY OF DURHAM

DEPARTMENT OF GEOGRAPHY

CENSUS RESEARCH UNIT

WORKING PAPER No. 4

JUNE 1975

STORAGE OF THE 1971 UK CENSUS DATA : SOME TECHNICAL CONSIDERATIONS

M. VISVALINGAM

Contents

STORAGE OF THE 1971 UK CENSUS DATA - SOME TECHNICAL CONSIDERATIONS

1. INTRODUCTION

Much thought has been spent on the design of a system for the storage, retrieval and processing of the grid-square based 1971 U.K. census data. This paper examines the need for a storage strategy and outlines some details pertaining to this strategy, which includes the selection of the 'nuts and bolts' or the constituent parts of the system through theoretical and experimental procedures. How these fit into the design of the ultimate information storage and retrieval system will be conveyed in a separate paper.

All experimental programming described herein was carried out in IBM Fortran G, and a minimum of Assembler routines. (Fortran was selected as it can be implemented in a machine-independent manner). Some utility routines available under the Michigan Terminal System (MTS) on the Northumbrian Universities Multiple Access Computer (NUMAC) were also used. At the time of the experiments, the Census Research Unit (CRU) of the Department of Geography, University of Durham, had simultaneous access to a maximum of two (of the four) 9-track magnetic tape drives and to a private 2314 disc pack (capacity 29 megabytes) plus considerable scratch space. These will be augemented by a single density 3330 disc pack with a storage capacity of 100 million bytes (or 100 megabytes).

This paper outlines a response to the local facilities and limitations of the available hardware, software and management at one installation. The experimental observations and time comparisons are undoubtedly affected by machine-dependent features. However, several of the considerations outlined and implications of findings are more general and basic.

2. NEED FOR STRATEGY

2.1 Volume of Data

The CRU has begun to receive 1971 UK census small area statistics, organised by 1 km grid squares, from the Census Division of the Office of Population Censuses and Surveys (OPCS). This will form the major data base for several research programmes such as analysis of population and spatial scale components and identification of demographic types and regions; in addition computer mapping will be carried out at national

and regional levels. Several practical problems arise from the sheer volume of data involved, even with the use of magnetic storage devices. The 100% population and household data for Durham census county comprises 920 variables for each of the 2041 1 km square units populated in April 1971. Unmodified, this occupies 1968 pages on an IBM 2314 disc pack (MTS writes data onto magnetic discs and drums in blocks of 4096 bytes called pages. In the context of sequential data files, the size of the file includes an overhead of 16 bytes for the file header and a further 6 bytes per record). In addition to the population and household data, some 651 socio-economic variables based on a 10 per cent sample of households are available for most grid squares.

Some 240,000 1 km squares exist within the UK; and on the basis of experience with the Durham data, data is expected for some 100,000 of these squares. Thus the total amount of data available for the grid-based units in the UK should be in excess of 600 million bytes - when received from OPCS and transformed into IBM computer 4-byte words.

In addition to the 1-kilometre data already described, summary totals for 100 kilometre blocks and in some areas a more detailed breakdown to 100 metre square units are also made available. The Census Research Unit is also using grid-referenced medical data provided by the Area Health Authority and it is likely that other data of this kind will become available. Problems of storage are aggravated by the need to keep at least one copy (preferably two) of the data as a stand-by should accidental corruption occur- especially since the original OPCS data files are in ICL code and have to be translated into IBM machine-readable form. Although these copies do not need to be on direct access devices, a significant tape library can result from simple storage of the data.

2.2 Consequences of the large volume of data

The entirety of the large volume of data is likely to span over several physically distinct storage media. Recent technological advances have produced magnetic media that are cheaper and of larger capacity. Even so, each has a finite storage capacity. At the time of writing the largest capacity disc normally available on IBM machines is the 3330, the double density version of which can hold 200

TABLE 1 : ANTICIPATED PERCENTAGE OF CALLS ON CENSUS GRID SQUARE DATA, FOR DIFFERENT SIZED DATA MATRICES

(Guesstimates by I.S.E.)

← AREA COVERED

NUMBER OF VARIABLES →

DIFFICULT	WHOLE UK	COUNTRY (Scotland, Ireland, England & Wales)	REGION or COUNTY 3, to 6,000 km^2	100 km^2	1 km^2	MULTIVARIATE FORMAT
ALL (1500)	-	-	1	1	1	3%
LARGE (200-300)	-	-	4	3	1	8%
MEDIUM (60-100)	-	2	10	3	1	16%
SELECT (15-30)	-	3	5	3	-	11%
2 or 3	-	5	5	1	-	11%
1 (including ratios) i.e. maps and spatial analysis	2	30	15	4	-	51%
	2%	40%	40%	15%	3%	100%

UNIVARIATE FORMAT

EASY

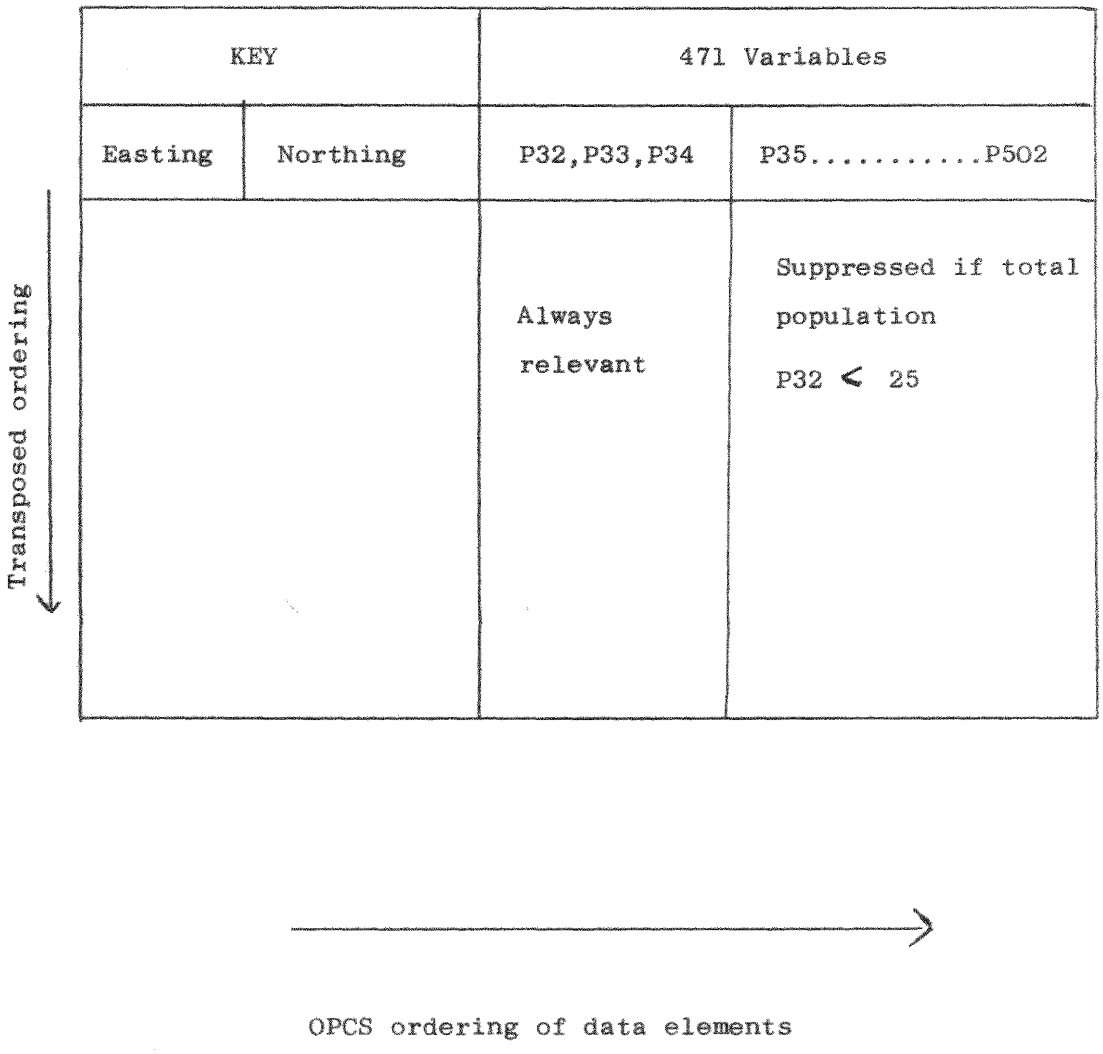
megabytes. Uncompressed,the 1971 census data for 1 km squares in UK would require at least three double density 3330 packs or at least 14 magnetic tapes (2400 feet, 9 track with a recording density of 1600 bytes per inch). Unfortunately, the mounting and dismounting of discs is still very time consuming, especially when the user is on-line.

Data usage patterns also vary considerably from sequential access of large parts of the data to the retrieval of relatively small subsets defined by areal or attribute or both criteria. 'Guessed estimates' of the frequency of access of various data subsets appear in Table 1. In the original data all population data for one grid square is stored in one record, followed by all household data for the square in the second record (this will be referred to as the multivariate format). The extraction of just one or two variables for the whole country would involve reading all the records from several physical units to use only a small fraction of the data actually transferred. Given that the location and transfer of data blocks are evaluated in milliseconds (while core processes are evaluated in microseconds) storage of data in its uncompacted form on several physical units can be extremely wasteful of store and time.

The large quantity of data and varying usage patterns also pose other problems in a virtual memory system (or paging environment). In such a system only those parts of the program and data (demarcated into pages) currently in use are held in fast core within the computer; the rest are automatically 'swapped' out onto fast backing store to be retrieved only when needed. Given the multivariate format described above, abstraction of attribute sets requires the continued access of non-adjacent elements. With a large data set this can induce a constant swapping of data pages with very little progress on the job in hand, creating a condition termed page thrashing.

The above problems can be alleviated by suitable file and data structures and by a process of data compaction. To this end experiments were conducted with the Durham population file to evaluate the degree of compaction feasible and the costs at which such economies can be made. The implications of transposing the file were also investigated, and these were related to the anticipated patterns of usage.

FIGURE 1 : STRUCTURE OF THE DURHAM 100 PERCENT POPULATION DATA



3. THE DURHAM CENSUS COUNTY POPULATION FILE

Data on 471 population variables is available for the 2041 one kilometre squares that were populated in April 1971 in Durham census county. These were made available by OPCS in 2041 fixed length records. When the total population (P32 in Figure 1) in the kilometre square area is less than 25, only figures for total males and females are made available in P33 and P34 respectively. The rest of the fixed length multivariate OPCS record then consists only of 'missing data'. The structure of the Durham population file is illustrated by Figure 1. The area key in Figure 1 is extracted from the 32-word record header given by OPCS and it identifies the kilometre square unit to which the rest of the record relates. With reference to Figure 1, the OPCS record with its multivariate-record format is thus a row-wise collection of data elements.

4. DATA COMPACTION

The object of data compaction is to retain the information in as few computer words as possible but phrased so that the compacted data can easily be restored into its original form. Compaction of data is desirable not only because compacted data occupies less space on backing store and hence relatively more data can be packed into one physical storage unit, but also because transfer of compacted data takes less time than the transfer of original data. There are many different strategies in data compaction. For example, each individual data element can be stored in a part of a computer word which is of just sufficient size to hold it, given the means of data representation within the computer and of manipulation by the particular programming languages being used. Data can also be transformed from one type of representation to another to save space allocation per element. If there is a preponderance of a particular value in the data, indexing the position and number of elements where it occurs may obviate the need to store the elements themselves. Substantial savings can also be made through an understanding of how various operating systems and software packages store data to enable selection of those procedures that are most advantageous to a given application. In addition, the choice of data structures and file organisation methods also determines the amount of space allocated to the data files.

In contrast to these methods of compaction, a process of data

culling (information reduction) could be employed to retain only relevant data. If the information requirements were predictable, some data could be discarded as being of marginal value, This paper is not concerned with data culling, though that is a valuable technique in many situations.

With many methods, savings in storage space can only be made at the cost of increased execution time and hence their desirability has to be carefully evaluated, initially with common sense and theoretical reasoning based on published comparisons and system specifications, but ultimately by experimental programming. The compaction techniques thus tested and those contemplated are outlined so that the cost - benefits of each may be examined.

4.1 Internal versus external representation

The bulk of the data from OPCS comes as unformatted full word binary integers, i.e. it is held on the external magnetic storage media in almost the same form that it takes within the computer. This representation was retained since it requires less storage space and less CPU time to read the data than does decimal representation.

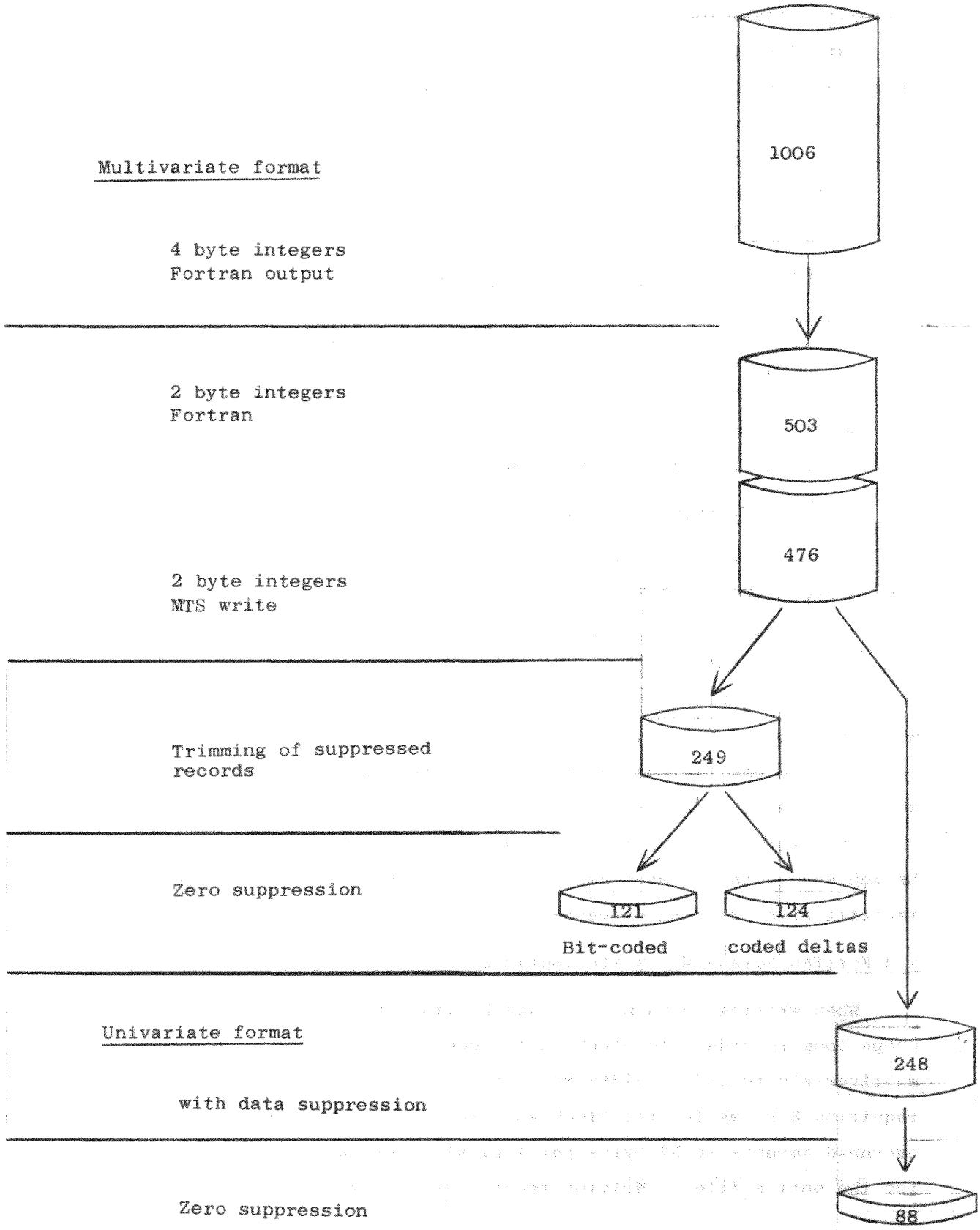
4.2 Change of Storage allocation per element

If the storage allocation is to be identical for all elements of the record, then the minimum amount of store must be chosen in relation to the maximum value for any one element. In the Durham County 100 percent population data, the maximum population per grid square is 10,386. It is highly unlikely that any other than summary values will ever exceed 32,767 (2^{15}-1) even in the national data. In view of this, use was made of the IBM FORTRAN facility to store non-summary data as INTEGER*2 (16 bits or 2 bytes), with an initial check being made of the input value. Some similar savings could be achieved with ICL machines using the COMPRESS INTEGER AND LOGICAL facility or by packing elements using PLAN routines.

4.3 Fortran versus MTS write routines

When writing to magnetic discs Fortran under MTS automatically chops long records into blocks not exceeding 255 bytes. Hence one multivariate record of 473*2=946 bytes is spanned over 4 blocks, each requiring 8 bytes for the block and record descriptor words. This overhead amounts to 32 bytes per logical record and 32*2041 bytes for the entire file. Writing records as contiguous blocks via

FIGURE 2 : COMPACTION OF THE DURHAM 100 PERCENT POPULATION FILE



an MTS output routine effected a change from 503 to 476 pages i.e. a saving of 27 pages (refer Figure 2).

4.4 Data suppression

Owing to confidentiality restrictions when the total population in a kilometre square is less than 25, only values for the total number of people and of males and females are provided, the rest of the information for the particular square being suppressed (refer Section 3). Hence the fixed length blocks of OPCS are very wasteful. Trimming off the redundant space in the suppressed records saves storage and transfer time. The complete removal of 49 records containing only zero values and the trimming of the 940 suppressed records saved 227 pages (Figure 2)

4.5 Blocking on magnetic tape

The process of trimming reduces suppressed records to a mere 10 bytes, each record containing x,y co-ordinates and the summary population data. With a recording density of 1600 bpi, the record would only occupy 0.01 inch and would be separated from the next by an inter-block gap of 0.56 inch. Hence one could quite easily end up with a situation of devoting more space to inter-block gaps than to data records. For this reason it is essential that several records are collected together and written as a single physical block. In most installations the user can specify the blocking factor to the operating system which would take care of the blocking and deblocking for the user.

4.6 Zero suppression

Space can also be conserved by suppressing preponderant values and keeping a note of their position and frequency of occurrence. To this end, indexing algorithms conceived for the storage and processing of large order sparse matrices were considered : Pooch and Nieder (1973) defined a sparse matrix as one possessing less than 15 to 25 percent non-zero elements. As the Durham population data is 36.5 percent dense, of the traditional methods they described only the bitmap scheme was likely to conserve space. In this scheme a map or pattern of 0's and 1's (to indicate zero and non-zero elements respectively) is stored together with a reduced vector of the non-zero values. For example the original array 0,0,3,0,12,0,0,0,104, 106, would be represented by:

bitmap : 0010100011

values : 3, 12, 104, 106

The position of the non-zero values is obtained by indexing sequentially through the bitmap. Hence the original array of N (here 10) elements is contracted to :

A (n) : array of n (here 4) non-zero value elements

and : B = $\frac{N}{b}$ elements for the bitmap,

where, b is the number of bits in the storage unit allocated to the element and

B is the smallest integer greater than or equal to $\frac{N}{b}$

(in the above example B = $\frac{10}{16}$ = 1 (2 byte element))

It is useful to store (n) as it minimises the number of tests needed. Hence the total requirement in this example is 6 elements for the compacted buffer. With the Durham population data $\frac{N}{b}$ +1 =31 elements. Since sequential bit indexing takes more time than indexing an uncompacted array, only those records that saved at least 200 bytes were compacted, which reduced the file down to 121 pages (see Figure 2)-saving a further 128 pages.

The position of the data item in the data list identifies the variable and the spatial unit it refers to : thus several applications demand that the data array be restored back to its original form for in-core processing. In this respect, the bitmap scheme was found to be rather slow, as the distribution of non-zero elements often requires that the entire bitmap be indexed. Reid (1974) has noted that such schemes may be much more efficient with parallel processors but, given the existing hardware, a method of run length coding (Rosenfeld, 1969) called the coded-delta scheme was examined and found to be substantially faster.

With reference to picture processing, Rosenfeld pointed out that when there is a tendency for one value to be more probable than all the others put together, the highly probable value tends to occur in 'runs' and the others in isolation. When this tendency is sufficiently strong he suggested that it may be more economical to encode the first value of each run and the length of the run, rather than encoding every value in sequence : this gives

rise to the term ' run length coding'. A similar scheme, 'address mapping', has been widely used in the indexing of sparse matrices; only non-zero values are stored together with their absolute position in the array. In large matrices, the co-ordinates defining the position of non-zero values are also likely to be large. The delta indexing scheme minimises these storage requirements by encoding with each non-zero value element a delta, which is the difference in position between the current and the preceding non-zero elements. (The difference, being smaller in value, requires a smaller unit of storage for the elements of the indexing vector than does the value of the absolute position). The position of the non-zero element is then determined from the sum of the preceding deltas. As the number of deltas or lengths is directly equal to the number of non-zero values, the method is very demanding on store when a matrix is not very sparse. With the Durham population data, possessing an average matrix density of 36.5 percent, these schemes require 129 indexing units for records with 473 elements. Hence the coded delta scheme was used in the hope of retaining a faster scheme of indexing while cutting down on core requirements.

In the coded delta scheme the need for an indexing vector is eliminated by encoding the position and number of zero values in the sequence in which they occur. The coded delta is a single value that gives the number of zero value elements separating two non-zero ones; and it is encoded in the reduced vector only when it replaces pre-existing zeroes. The delta is encoded in its correct place within the array of values by using a single bit to differentiate it from true matrix values. As the Census data consists entirely of counts and ratios taking zero or positive values, the topmost (sign) bit can be used to code a delta, which thus takes the form of a negative number. For example, the array of values in the previous example can be coded as :

-2 , 3 , -1 , 12 , -3 , 104 , 106

Although the storage requirements of the original array are never exceeded, no matter how dense the matrix, the savings made depend entirely on the number of intervening blocks of zero values. On storage, its greatest compaction potential is when a few large blocks of zero values exist in the data list. Where the space needed for the total number of deltas exceeds that required for

TABLE 2 : CPU TIME TO REGENERATE RECORDS AT DIFFERENT LEVELS OF COMPACTION

		ENTIRE FILE		MTS READ OF DATA FILE; FORTRAN READ OF INDEX TABLES			To read one variable using MTS read
		Fortran unformatted Read	MTS Read	with Data Suppression	Zero Suppression		
					Bit-coded	Negative Deltas	
OPCS FORMAT	Size of file (pages)	476		249	121	124	476
	CPU time (seconds)	30.1	4.5	11.4	26.2	18.4	4.5
TRANSPOSED	Size of file (pages)	248			88		248
	CPU time (seconds)	15.3	1.8		17		0.1

2041 records

474 records

the bitmap, it becomes progressively less efficient in terms of space compaction. Table 2 illustrates that, with the Durham population data, the negative delta method was less economical than bit mapping but it is a much more rapid method of indexing. The compaction achieved with negative deltas can be maximised by re-organising the columns of the data matrix such that those columns (variables) containing large numbers of zeros are adjacent to each other.

Other methods of zero suppression (refer Page and Wilson(1973), Pooch and Nieder (1973) Tewarson (1973), Reid (1974)) may be more suited to some other types of applications and computing environments.

4.7 Transformations

While the Durham census data can be represented as half-word integers, it is clear that some of the user-derived variables such as very small proportions and percentages are most easily represented as reals. Equally, summary totals will exceed the capacity of two-byte storage. Most unpredictably, values in the 'raw' census data may also grossly exceed the capacity of two -byte storage owing to the adjustment which is carried out for confidentiality reasons (Rhind 1975). Such variety in the data not only requires additional codes to indicate the type of representation involved but also automatically increases requirements unless the variables are created with some forethought.

If the new variable is a straightforward transformation of an existing one, for example, its square root, reciprocal etc., then it is more economical in space if this is derived at run time rather than storing both old and new values. More often a variable is derived via a complex function involving several other variables. Hence it is expedient to retain the derived variable not only to economise on CPU time, but also to save the user the tedium of re-specifying it each time, even though this increases the store required. Core requirements can be minimised if percentages are held as half-word integers, representing ratios as 'per 10,000' , or other similar adjustments are made, depending on the accuracy required. Changing the definition of variables may also be helpful e.g. changing sex ratio from (males/females) to (males/total population) limits the range of the values from 0 to 100, rather than from 0 to plus infinity (it also has interpretational advantages). In some univariate analyses the derived variable is equally useful when standardised

to zero mean and unit variance.

Retention of certain summary statistics like the mean, minimum, maximum, standard deviation and other user-defined parameters which are likely to be of frequent use in several applications will reduce CPU time and may save extra passes through the file, thus compensating for the small increase in storage requirements.

4.8 Data ordering

As most users of census data require many different categories of data for the same areal coverage the multivariate format adopted by OPCS is ideal for the collation and distribution of such data. Each data record gives the values for a list of variables aggregated in one spatial division. There are several applications for which data elements are better transposed and stored by variable than by location (refer Section 5). Such transposed files with a univariate format can be held much more compactly than the multivariate - records. At the final stage of compaction the Durham population data occupied 124 and 88 pages respectively in the multivariate - and univariate (transposed) record formats (refer Figure 2). The reasons for this are discussed more fully in Section 5.

4.9.File organisation

Economies in store can also be made by a selective use of file types offered by the operating system. Files on magnetic tapes are somewhat restricted and once the contents and data structures have been carefully tailored, the one further economy that can be effected is via blocking (refer Section 4.5). Files on direct access storage devices are, on the whole, much more versatile. MTS under NUMAC offers the user only two types of file structure, namely line and sequential files. Line files have an indexed sequential method of organisation and are extremely useful when information needs to be accessed in a random manner with keys (which map to line numbers). MTS line files unfortunately have several restrictions which make them unsuitable for large data files. The size of the file is limited to a fiftieth of a disc pack's capacity. Moreover their method of organisation requires both a record index and a line directory for random access. Lines are restricted to a maximum length of 255 bytes and while some software packages (including Fortran unformatted input/output) segment a long record into several lines, this involves considerably more storage overheads,

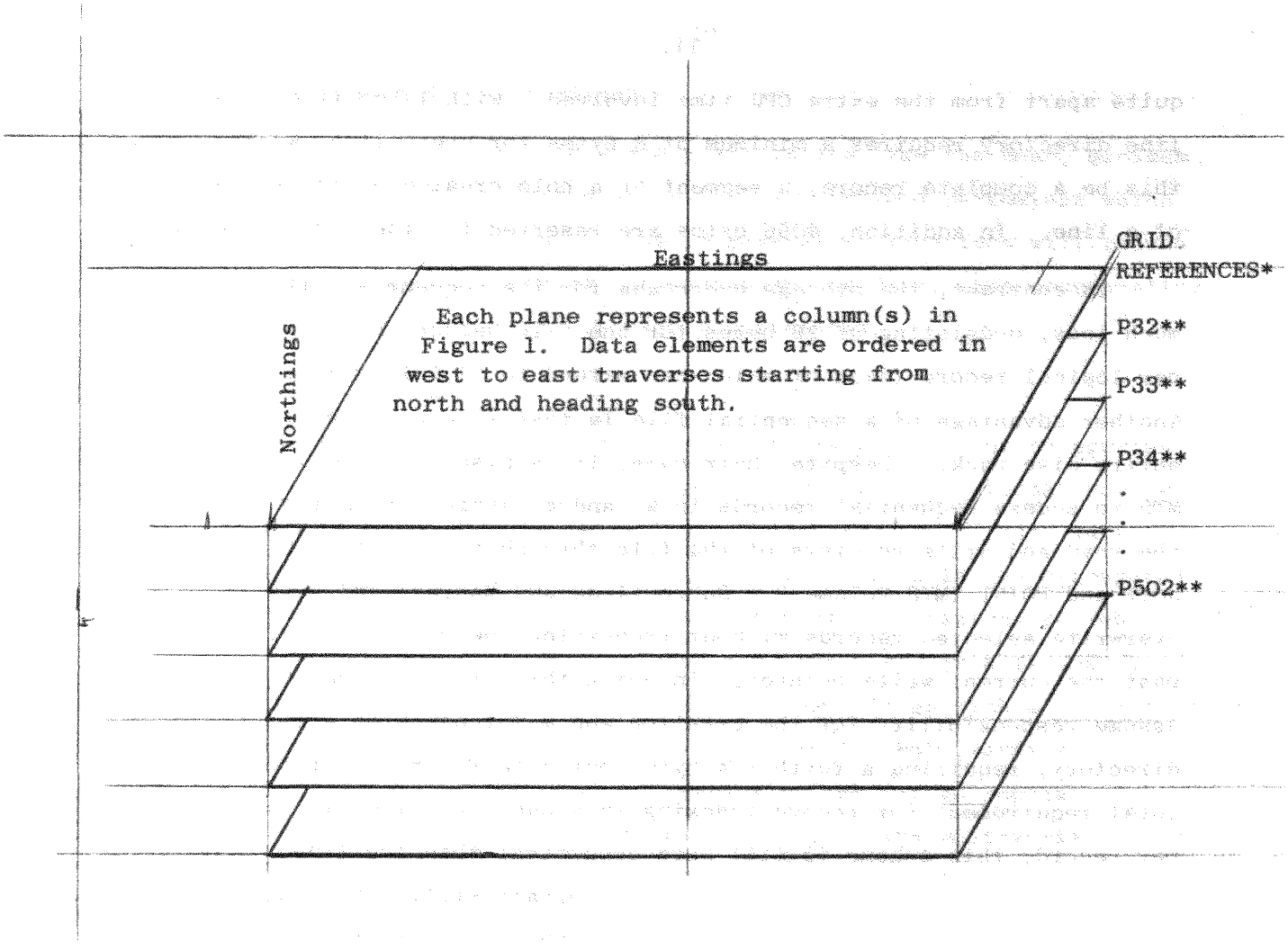
quite apart from the extra CPU time involved : with large files, the line directory requires a minimum of 8 bytes for every line, whether this be a complete record, a segment or a hole created by the removal of a line. In addition, 4096 bytes are reserved for the record index.

In contrast, the storage overheads for the sequential file are much less, consisting of 16 bytes for the file header and 6 bytes per logical record (each of which can extend to 32,767 bytes). Another advantage of a sequential file is that it can occupy an entire disc pack. Despite their name, it is also possible under MTS to access sequential records in a random fashion by manipulating the read and write pointers of the file through the system subroutines NOTE and POINT (MTS Volume 3 - Subroutines and Macros) and even to overwrite selected records without truncating the rest of the file past the current write pointer. In doing this, the user has to assume responsibility for the creation and maintenance of a record directory, requiring a further 4 bytes per data record. Although the total requirement for record indexing in sequential files is larger per record, this scheme is still more economical than the line directory for long records. But more important still, the index table can be held in a separate file, together with other organisational information, producing a much more versatile system. An outcome of this arrangement is that the non-existence or suppression of a particular record can be ascertained without having to open the main data files. Moreover, it enables the user to construct secondary file structures (Cardenas, 1975) which are optimal for retrieval of subsets of the data body.

Hence, use of the sequential files is the obvious choice for our data files, while the various index tables are more conveniently held as line files.

5. DATA ORDERING

While individual elements are easily accessed and processed in the main memory, data is transferred to backing store to and from buffers in the main memory of the computer in blocks of data elements. This requires that the data body be segmented or organised into collections of data items in some rational manner. In the multivariate format a data record is a row-wise (with reference to Figure 1) collection of data elements. The



* Data for grid references consists of a vector which identifies the spatial units for which data is available.

** P32 to P502 are univariate records which contain a list of values corresponding to the above locations

FIGURE 3 : ANOTHER REPRESENTATION OF THE DURHAM 100% POPULATION DATA

juxtaposition of different variables for each kilometre square unit expedites the collation of data for specified areas. Figure 3 illustrates the way a geographer commonly views spatial data, reflecting his traditional reference to the map as the data storage model. The information content is the same as in the multivariate format, but the difference in view though subtle is significant. Automated data processing has to consider this and the associated usage patterns in its storage model and strategy to conserve user time and computer resources. The view represented by Figure 3 requires that the data be stored as attribute or univariate sets, i.e. in column order with reference to Figure 1. After compaction copies of the Durham population data in both formats only required ((88+124)/1006x100)= 22 percent of the original store requirements. If the same degree of compaction could be achieved with the UK data it may be possible to store the data in both formats. Even when the system can cope with either approach, a careful choice of the correct format (data file) can yield substantial savings in execution and elapsed time : in an on-line context this means savings in response and hence ultimately in user time. Some implications of the alternative schemes of data ordering are outlined below : these should be considered where available resources require that a choice be made.

5.1 Physical considerations

5.1.1 Storage Unit capacity

With technological advancements data storage media have become more sophisticated. Large data sets have progressively migrated from bulky and clumsy punched cards and paper tapes on to magnetic media such as tapes, discs, drums and data cells of various specifications. Even so each has a finite storage capacity, which is either explicitly quoted (for example the 2314 and double-density 3330 have a capacity of 29 and 200 megabytes respectively), or can be calculated from other details available. For example given a 9-track tape of 2400 feet and a recording density of 1600 bpi, each tape has a maximum capacity of :

1600 x 12 x 2400 = 46 million bytes

These figures for maximum capacity do not include physical space used by organisational information, inserted by the operating system for its benefit, or by inter-block gaps that separate the data blocks. When the data blocks are relatively small, the latter can

occupy as much or even more than half the capacity of the storage device. Even disregarding the additional storage overheads, the 1971 UK census data certainly will not fit into any one of these storage units in its original form. As computer operations are reckoned in micro-and milliseconds, the mounting and dismounting of storage devices by human operators can waste a lot of terminal time. The aim of data compaction and data ordering is to ensure that the required information can be obtained by scanning a few units, preferably only one.

OPCS grid-square data for the UK contain some 100,000 records each consisting of 473 2-byte elements in the hundred percent population file. The maximum number of such records that can be held on a 2314 disc pack is :

$$\frac{c}{b \times e} = \frac{29{,}000{,}000}{2 \times 473} = 30{,}000 \text{ records}$$

whre, c = capacity of unit in bytes
b = number of bytes per element
e = number of elements.

This implies that even for population characteristics alone, information on a 2314 disc pack is only complete for subareas extending over 30,000 spatial units. If an attribute set for the whole country is required, several storage units will have to be mounted. Table 1 clearly suggests that the data is more often required for large areas but for relatively a small number of variables. Transposing the 1971 UK census data produces attribute sets; the maximum number of such univariate records that can reside on the 2314 is :

$$\frac{c - (n_f . e_f . b)}{e_s . b} + n_f = \frac{29{,}000{,}000 - (5 \times 100{,}000 \times 2)}{60{,}000 \times 2} + 5$$

$$= \frac{28{,}000{,}000}{120{,}000} + 5 = 238 \text{ records}$$

where, n_f = number of full records
e_f = number of elements in above
e_s = number of elements in suppressed records.

TABLE 3 : FREQUENCY DISTRIBUTION OF DURHAM 100% POPULATION RECORDS

- WHERE CLASS INTERVALS ARE RELATED TO CORE UNITS

	Bit Coded		Coded Deltas	
Core Unit	allocation of record max.	frequency	allocation of record max.	frequency
2 bytes	256-32767	118	128-32767	163
1 byte	16- 255	(257	8- 127	(259
		342(		297(
hexadecimal	1- 15	(85	1- 7	(38
Ø	Ø	13	Ø	13
		TOTAL: 473		TOTAL: 473

At maximum 238 attributes can be held for the whole UK, on one 2314 which can therefore be used for most requirements. Although the compaction process allows a greater number of records of either type to reside on the 2314, the above example illustrates one type of choice that data ordering involves.

5.1.2 Transfer time

To extract data for just one variable with the multivariate format all records need to be transferred, and this involves several storage units. Since only 0.2 percent of the data transferred is used, this process is very wasteful. The univariate format ensures that only the relevant records need to be transferred when the data is being accessed as attribute sets. The execution (CPU) time quoted in Table 2 includes only the time taken for initiating and checking transfers. The total transfer takes a much larger proportion of time, logged under elapsed time; this is difficult to ascertain accurately in a multiple-user environment. While core processes are evaluated in microseconds, the location and transfer of data blocks is evaluated in milliseconds; hence the absolute values in Table 2 grossly under-estimate response time. In multivariate format, a retrieval of one or all variable(s) from a full 2314 disc necessitates reading the whole disc, a process absorbing 287.1 seconds of retrieval time (Appendix 1). From Section 5. 1.1 above this represents a sequential read of the 30,000 population records in multivariate format. In univariate format it takes N x 1.18 seconds to read N attribute sets. Both measured CPU time (Table 2) and estimated retrieval times show that transposed ordering of data elements is more efficient for the usage pattern figured in Table 1. Terjanian and Lefebvre (1972) adopted a similar storage scheme for data files containing around 100 attributes for each of 22 million individuals on the reasoning that no more than 6 variables would be required at a time for most purposes.

Measured CPU time comparisons also indicate that it is faster to read the entire Durham hundred percent population file in univariate format. This is because the data is collected into a fewer but longer records (refer Table 2). While the transfer rate remains the same, the initiation process which includes the location of the record on the direct access storage device (DASD) is invoked less often. The difference in time taken to read the entire UK

data on population characteristics is likely to be even larger, as the number of spatial units and hence multivariate records will increase to approximately 100,000 while the number of attributes in the population file will still be 471. No doubt the univariate records themselves would be segmented to process subareas more efficiently. However, even if each of these were segmented into 1 page blocks, less than 15,000 records would be created.

5.1.3 Storage requirements

The experimental results from the Durham data also confirm that univariate sets can be held much more compactly than multivariate sets as apparent from Table 2. This is due to several reasons. Firstly, both types of file organisation for DASD offered by MTS involve storage overheads that are directly proportional to the number of records; on magnetic tape the fewer longer univariate sets have also to accommodate less inter-block gaps.

Secondly, 49 out of the 2041 multivariate records and 13 out of the 473 variables (especially those univariate records relating to married males or females between the ages of 0 and 14) consisted entirely of zero values and were not stored. This again represents a greater saving in the univariate scheme.

Thirdly, transposing has the effect of grouping items of like magnitude together, which facilitates a much greater degree of compaction. Table 3 shows that the majority of univariate records contain elements that fit conveniently into a single byte. Files with multivariate and univariate record formats occupied 124 and 88 pages respectively after the final stage of compaction (refer Table 2).

((The use of single bytes for some univariate sets requires that the methods adopted for zero suppression be reconsidered. The IBM FORTRAN G compiler cannot perform arithmetic operations on storage units smaller than halfword size. Hence values packed into bytes have to be restored to halfword form before any further arithmetic operations can be performed. If the element contains an item of census data then the topmost byte of the halfword has to be cleared to zero. With the coded delta scheme, if the packed element is a delta, then the topmost byte of the halfword must contain the value -1 (or hexadecimal 'FF') as illustrated below:

packed	restored	value
00000010	0000000000000010	2
11111110	1111111111111110	-2

With the bitmap scheme all values in the reduced vector are positive (refer Section 4) : thus the topmost byte of the halfword is always zero. This avoids the need to test the state of the packed element or to change the content of the topmost byte. Hence the performance of both schemes was evaluated again using attribute sets that could be packed into one-byte arrays. An array of 1052 halfword elements, containing only 255 non-zero values was compacted by the bitmap and coded delta schemes to 395 and 404 bytes respectively: it took 41 and 18 seconds respectively to read and regenerate the above reduced arrays a thousand times. A similar test with an array of 1052 halfword elements, containing 974 non-zero values took 75 seconds by the bitmap scheme but only 35 seconds for the coded delta method. Hence the coded delta scheme was more desirable.

However, mixed methods of zero suppression were adopted for attribute sets for the following reasons. In some records the number of blocks of zero was considerably in excess of the requirements of the bitmap. Moreover, as the census data consists of counts, the maximum value that can be held in a byte is $2^8 - 1 = 255$.

As the bitmap scheme stores only the positive non zero values, it can store a maximum value of 255. The coded delta scheme, however, reserves one-bit to differentiate between values and deltas. Thus the largest integer that can be stored in a byte is $2^{(8-1)} - 1 = 127$. Hence when the maximum value in a record is between 127 and 255, the coded delta scheme requires 2 bytes for each element while the bitmap scheme only requires one. The tentative packing scheme, using mixed modes of compaction on records which saved at least 200 bytes, resulted in a file of 88 pages (refer Figure 2) and it took 17 seconds (Table 2) to read the compacted file and regenerate the original set of records)).

The univariate format requires less space for user-created index tables for random access, as again the number of entries in such tables is directly related to the number of records in the file. For the UK data a file in multivariate format requires (n.r) entries (where n= number of locations and r= the number of record types, currently 4) which would be approximately 400,000 entries. On the

TABLE 4 : EFFECTS OF NUMBER OF VARIABLES TRANSPOSED ON RESPONSE TIME

Number of Variables	array* size	CPU time (seconds)	‡ elapsed time (minutes)	‡ maximum virtual memory requirement	‡ drum** reads	‡ response time at terminal
100	B(473) A(2041,100)	34.0	17.80	120	19,659	2 hours (at tea time - low demand)
50	B(473) A(2041,50)	31.9	2.05	70	358	25 minutes (peak demand period)
1	B(473) A(2041)	30.2	-	34	269	20 minutes (peak period)

* B(473) is the buffer into which the multivariate records are transferred
A(2041) is the array of data for one univariate record (or one 'map' in Figure 3)
A(2041,n) is the array of data for n univariate records

** drum reads indicate the degree of page swapping induced

‡ Although these figures are unstable and depend on concurrent demands on the system, the relative difference in values are of concern

other hand, for a file in univariate format, the number of entries equals the number of variables, altogether 1571.

5.1.4 Linear core and page swapping

The format adopted has also a direct bearing on the efficiency of in-core processing. In a virtual memory system, only the parts of the program and data (demarcated into pages) currently in use are held in core store, the rest are automatically swapped out onto fast backing store to be retrieved only when needed. Digital computers moreover are organised to receive and process data in a one-dimensional sequence. Records are transferred from the data files into the linear set of storage units allocated to arrays in the same sequential data order that they occupy on external storage media. Hence in Fortran, where the left-most subscript changes most rapidly, the multivariate format requires that the array (A), with spatial (s) - and attribute (a) -dimensions, be declared as A(a,s). In the univariate format, however, the array declaration will take the form A(s,a). Hence with large arrays if the usage pattern requires the access of non-adjacent elements, it can induce a constant swapping of data pages with very little progress on the job in hand. The process of transposing the OPCS file for Durham census county induced a similar condition as illustrated by the figures for elapsed time and drum reads on Table 4. In the IBM 360/67 with the size of dimension (s) being 2041, a serious condition of page thrashing develops when (a) exceeds 75 variables. Page thrashing is affected also by concurrent activities in the system and other installation features and hence the threshold figure would have to be ascertained experimentally. Once the threshold is exceeded a mismatch between the physical data ordering and the indexing pattern can affect response time very drastically With grid based data, one is concerned with three (two spatial and the attribute) dimensions. Just as there is a choice between multivariate and univariate sets, there are also similar considerations with respect to latitudinal versus longitudinal ordering of grid cells.

5.2 User manipulations

5.2.1. Addition of new information

Any data base must be capable of systematic expansion to include new attributes, including user-derived variables and extentions in areal coverage. The incorporation process must check that already

existing information is not duplicated and that such additions do not invalidate existing index tables. The addition of new variables into the univariate format does not alter the structure of the existing file or the contents of the already existing index tables greatly but it merely extends the body of information in a totally compatible manner, i.e. it merely involves the addition of further columns to Figure 1 or layers to Figure 3. Pre-existing attributes need not be stored but are recognised as synonyms in the index tables.

With the multivariate format, the incorporation of new grid-based information, or even of new variables specified by the user from the basic OPCS data is more disruptive. There are three chief ways in which this can take place. New information pertaining to an already existing spatial unit can be tagged onto the end of the current record. This is clumsy because information may not relate to exactly the same area or to grid-squares of the same size. Furthermore the various OPCS record type (100% population, and 100% household data) are suppressed individually, which makes the compaction of merged records more awkward. This method of merging would not only increase core requirements for the buffer area, but is also likely to involve the transfer of a greater proportion of irrelevant data in some situations.

Secondly, the new records can be merged into the old files, interleaved between existing records and relating to the same spatial units. Although this facilitates the derivation of new variables because all the information pertaining to a spatial unit is physically grouped together, it means a complete rearrangement of the existing data files and index tables.

The new collection of data can be appended as a separate file and this is by far the least disruptive process. However it is still not as convenient as the univariate format as new indexing information would still have to be merged with already existing index tables for more economic storage and to facilitate cross-referencing between the different record-types and files.

While the univariate format is more conducive to the addition of new attributes, the multivariate format is more convenient for extending the areal coverage, as each record identifies the spatial unit explicitly by a key. When data records are required in the defined areal sequence, this requires that the new records be slotted in position or that the records are read in a non-sequential

fashion. The univariate format on the other hand requires that a similar process be adopted for elements within each attribute set. The attribute sets for the new area can be stored separately and chained to the existing lists. Conceptually this represents an overlay onto the existing layer in Figure 3. However, if there were several insertions into the already sorted univariate records, it is more economical in the long run to incorporate the new elements into the already existing raster order, which involves reshuffling the contents of every record in the file.

Thus in a dynamic data base the choice between the multivariate and univariate format is also conditioned by the relative stability of the area- and attribute dimensions.

5.2.2 The derivation process

Manipulation of spatially independent data commonly involves the formation of new attributes, or extraction of subsets by multiple attribute keys. An important further question with spatial data is the derivation of compound spatial units, so that spatial variation can be assessed at different areal scales.

Derivation of new attributes from primary ones and extraction of subsets by attribute keys is more easily done with the multivariate format. Here the primary attributes are all physically close together in core and require a smaller buffer as the information is processed a row at a time. With the univariate format however, it is convenient to hold the entire array of values for each primary attribute in core throughout the process of derivation. With the Durham set this was not a serious handicap so long as the number of primary attributes under consideration was not greater than about 75 (refer Section 5.1.4). For similar reasons, the univariate format is more conducive to spatial processing, including mapping, the determination of population potential, spectral analysis etc.

5.2.3 Input to packages

Various statistical and survey analysis packages, like ECXP, XDS3, SPSS etc., require that data be prepared in a format that reflects the OPCS multivariate ordering of data elements. Hence, it is much easier to abstract information from the data base and preprocess it for these packages if the multivariate format was retained (refer Section 5.1.4). However, several of these packages have capabilities for retaining the original as well as derived

(correlation, cross product etc.) matrices to suit their system specifications. The univariate format can pre-process about 75 variables in the Durham data without having to use intermediate work files to re-format the data.

6. FILE ORGANISATION

While records collect together the raw items or elements of data in a manner that facilitates the transfer and processing of these elements, the arrangement and interrelation of records in a file depend upon the manner in which the records are to be processed. The MTS sequential file organisation was adopted for the data files for reasons outlined in Section 4.9. In the OPCS-type file, the multivariate records occur in west to east order, proceeding from north to south. However, when records referring to subareas are required, the logical order in which the records are required no longer corresponds to the physical order in which they are stored. Hence, to avoid reading a large part of the file to abstract data for a small area, it is expedient to store pointers to these records along with keys that identify them in a direct access table. However, the storage of all potential keys and associated addresses may occupy several pages. With the UK data approximately 400,000 fullword entries may be required (4 types of records relating to approximately 100,000 locations) and these would occupy close to 400 pages on magnetic discs. Hence several methods of secondary file organisation have evolved to suit varying requirements (refer Lefkovitz, 1969). Experiments with alternative file structures for the Durham data is underway. The methods being investigated attempt to provide means of ascertaining the existence of, and locating individual records rapidly without having to store a pointer to each individual record. An alternative logical order of retrieving records, namely from north to south, proceeding from west to east together with backward indexing capabilities is also desirable. A rapid means of 'chasing' records in forward and backward directions both latitudinally and longitudinally is essential for problems which involve aggregation based on the values of given attributes; for example, for converting from constant areal to constant population units.

The organisation of the map formatted file is simpler as there are only 1571 entries, which do not need to be linked to each other in any particular order. The details of experiments relating to

file organisation methods will be presented in a later paper but it must be appreciated that this is another major consideration of a data storage and retrieval system in bridging the gap between the logical user requirements and the physical realities of storage media and computer hardware.

7. CONCLUSIONS

The need for a storage strategy arises from the sheer volume of the UK census data, which is considerably in excess of the storage capacity of currently available units of storage media. Data compaction techniques were used not only to conserve on storage space but also to enhance response time such that the solution of complex geographical problems, using large data sets, becomes feasible in practice as well as in theory. Some of the economies made are machine-dependent and have to be a reponse to the facilities and limitations of hardware, software and management at the user's disposal. At the other extreme are economy measures that are totally conditioned by the inherent structure and characteristics of the data at hand. Between these two extremes, however, are other more general methods which can be applied to data sets in almost any computing environment - the trimming of data elements, blocking, zero suppression, data ordering and transformations are some examples. Some compaction techniques can again be decided in advance while others (like zero suppression) have to investigate the actual values themselves. There is no ready-made compaction programme as such that is universally applicable to all data sets and this paper has summarised some of the sensible and available techniques.

Data compaction saves storage space and thereby transfer time. Further economies in time and store can be made by careful selection of file and data structures to ensure that data records are located speedily and that a relatively small amount of irrelevant data is actually transferred. The order of data elements (access paths) must reflect data usage patterns so that page swapping is minimised. In this respect the user has to ascertain whether the study is 'more' concerned with a small subset of areas or a small number of attributes; the data formats required by available package programs form an additional consideration. Moreover, in an expanding data base the relative stability of the different data dimensions has to be anticipated. The choice of data ordering

with respect to the technical considerations of storage requirements, transfer time and capacity of storage media is relatively easy and can be computed given the characteristics of the data base and file storage devices.

If the UK data can be subjected to the same degree of compaction as the population file for Durham census county, a copy of the UK data in both the multivariate and univariate formats can be stored on the available 2314 and expected 3330 disc packs. The tremendous improvement in response time would justify the duplication of data.

The brief discussion of motives for constructing secondary file structures stresses that the performance of a data base is dependent on the match between user requirements and the technical details of storage organisation. This paper has outlined some of the practical considerations involved in the detailed mapping of a data set onto storage media, and has identified some aspects of the usage patterns that are of relevance in designing an information storage and retrieval system.

ACKNOWLEDGEMENTS

I am extremely grateful to Mr. J.S. Roper of the Computer Unit for giving me a quick and comprehensive introduction to the use of direct access storage devices and file structures under MTS; and to the other members of the Census Research Unit, especially Drs. I.S. Evans and D.W. Rhind, for help with documentation, but accept full responsibility for the views expressed. It is my pleasure to thank Mrs. J. Dresser, who patiently typed the several drafts and manuscript and the Drawing Office for preparing the illustrations.

REFERENCES

CARDENAS, A.F. Analysis and performance of inverted date base structures, Comm. ACM, Vol. 18 (5), May 1975, p. 253 -263.

LEFKOVITZ., D., File structures for on-line systems, Spartan Books, New York, 1969

PAGE, E.S. and WILSON, L.B., Information representation and manipulation in a computer, Cambridge University Press, Cambridge, 1973.

POOCH, W.W. and NIEDER, A., A Survey of indexing techniques for sparse matrices, Comput. Surv., Vol.5 (2), June 1973, p. 109-133.

REID, J.K., Direct methods for sparse matrices, in 'Software for Numerical Mathematics" (ed. D.J. Evans), Academic Press, London, 1974, p. 29-47.

RHIND, D.W., "Geographical analysis and mapping of the 1971 UK Census Data", Working Paper No.3, Census Research Unit, Department of Geography, University of Durham, 1975

ROSENFELD, A., Picture processing by computer, Academic Press, New York, 1969, p.16.

TERJANIAN, A.S.R. and LEFEBVRE, J.J., Le géocodage de Statistique Canada : système d'information statistique géographique automatisé au stade opérationnel, in"International Geography 1972, Vol. 2" (ed. Adams, P.W. and Helleiner, F.M)., 1972, University of Toronto Press, 1972, p. 989-991.

TEWARSON, R.P., Sparse Matrices, "Mathematics in Science and Engineering" (series ed. Bellman, R.), Vol. 99, Academic Press, London, 1973.

APPENDIX 1

RETRIEVAL TIME FOR READING ATTRIBUTE SETS IN MULTIVARIATE AND UNIVARIATE FORMATS

MULTIVARIATE FORMAT

This involves reading a full 2314 pack, consisting of 200 cylinders. Each cylinder contains 20 tracks, each of which is capable of storing 7294 bytes. When data for the whole of UK is to be stored, a single disc pack can only hold 238 attribute sets at maximum (refer Section 5.1.1). The extraction of attribute sets involves a sequential read of the entire disc pack - and as all data would have to be transferred for any number of attribute sets, the minimum and average retrieval times correspond to the maximum.

Retrieval time = S + R + T

where S = total seek time = nc x s

R = total rotational delay = nc x p x r

T = total transfer time = t x c

nc = number of cylinders

s = average seek time

p = pages/cylinder

r = average rotational delay

t = average transfer rate

c = capacity of the disc pack

$\therefore$ Retrieval time = $(200 \times 60) + \left(200 \times \frac{145,880}{4096} \times 25.5\right)$

$+ (0.00320 \times 29,170,000)$ msec

$= (200 \times 60) + (200 \times 35.64 \times 25.5) + 93,344$ msec

$= 200 \times 968.82 + 93,344$ msec

$= 287.108$ sec

UNIVARIATE FORMAT

This requires access to the start of a variable and then a sequential read of the attribute set.

Retrieval time = N(cv x s + pv + r + bv x t)

where N = number of variables required

cv = cylinders per variable

pv = pages per variable

bv = bytes per variable

Assuming that the number of bytes per variable is 120,000

$$\text{Retrieval time} = N\left(\frac{120{,}000}{145{,}880} \times 60 + \frac{120{,}000}{4096} \times 25.5 + 120{,}000 \times 0.00320\right) \text{ msec}$$

$$= N(49 + 747 + 384) \text{ msec}$$

$$= N(1.180) \text{ sec}$$

Hence, the minimum time 1.18 seconds to retrieve data for one variable: the univariate format is thus much more efficient for the study of a small number of variables at a time.